**Étape 2A : Création du fichier de validation NIP**

/\*\*

\* =============================================

\* 🔧 VALIDATION NIP FORMAT XX-QQQQ-YYYYMMDD

\* Fichier: public/js/nip-validation.js

\* =============================================

\*/

// Configuration globale

const NIP\_CONFIG = {

format: 'XX-QQQQ-YYYYMMDD',

regex: /^[A-Z0-9]{2}-[0-9]{4}-[0-9]{8}$/,

maxLength: 16,

example: 'A1-2345-19901225',

apiUrl: '/api/v1/validate-nip'

};

/\*\*

\* ✅ VALIDATION FORMAT NIP CÔTÉ CLIENT

\* @param {string} nip

\* @returns {object} Résultat de validation

\*/

function validateNipFormat(nip) {

if (!nip || typeof nip !== 'string') {

return {

valid: false,

message: 'NIP requis'

};

}

// Nettoyer le NIP (supprimer espaces)

nip = nip.trim().toUpperCase();

// Vérification regex de base

if (!NIP\_CONFIG.regex.test(nip)) {

return {

valid: false,

message: `Format invalide. Attendu: ${NIP\_CONFIG.format}`,

example: NIP\_CONFIG.example,

help: 'Format: XX-QQQQ-YYYYMMDD (ex: A1-2345-19901225)'

};

}

// Extraction des parties

const parts = nip.split('-');

const prefix = parts[0]; // XX

const sequence = parts[1]; // QQQQ

const dateStr = parts[2]; // YYYYMMDD

// Validation date

const year = parseInt(dateStr.substring(0, 4));

const month = parseInt(dateStr.substring(4, 6));

const day = parseInt(dateStr.substring(6, 8));

// Vérifier si la date est valide

const date = new Date(year, month - 1, day);

if (date.getFullYear() !== year ||

date.getMonth() !== month - 1 ||

date.getDate() !== day) {

return {

valid: false,

message: 'Date de naissance invalide dans le NIP',

extracted\_date: `${year}-${month.toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`

};

}

// Validation plage d'années

if (year < 1900 || year > 2100) {

return {

valid: false,

message: 'Année de naissance invalide (doit être entre 1900 et 2100)'

};

}

// Calculer l'âge

const today = new Date();

const birthDate = new Date(year, month - 1, day);

const age = Math.floor((today - birthDate) / (365.25 \* 24 \* 60 \* 60 \* 1000));

// Validation âge

if (age < 18) {

return {

valid: false,

message: `Personne mineure détectée (âge: ${age} ans). Seules les personnes majeures sont autorisées.`,

age: age,

errorCode: 'UNDERAGE'

};

}

if (age > 100) {

return {

valid: true,

message: `Âge suspect détecté (${age} ans). Veuillez vérifier.`,

warning: true,

age: age

};

}

return {

valid: true,

message: 'NIP valide',

extracted\_info: {

prefix: prefix,

sequence: sequence,

birth\_date: `${year}-${month.toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`,

age: age

}

};

}

/\*\*

\* ✅ FORMATAGE AUTOMATIQUE DU NIP PENDANT LA SAISIE

\* @param {string} input

\* @returns {string} NIP formaté

\*/

function formatNipInput(input) {

// Supprimer tout ce qui n'est pas alphanumérique

let value = input.toUpperCase().replace(/[^A-Z0-9]/g, '');

// Appliquer le format XX-QQQQ-YYYYMMDD

if (value.length > 2) {

value = value.substring(0, 2) + '-' + value.substring(2);

}

if (value.length > 7) {

value = value.substring(0, 7) + '-' + value.substring(7);

}

if (value.length > 16) {

value = value.substring(0, 16);

}

return value;

}

/\*\*

\* ✅ INITIALISER LA VALIDATION NIP SUR LES CHAMPS

\*/

function initNipValidation() {

const nipInputs = document.querySelectorAll('input[data-validate="nip"], input[name\*="nip"]');

nipInputs.forEach(input => {

// Formatage automatique pendant la saisie

input.addEventListener('input', function(e) {

const formatted = formatNipInput(e.target.value);

e.target.value = formatted;

// Validation en temps réel si format complet

if (formatted.length === 16) { // XX-QQQQ-YYYYMMDD = 16 caractères

validateNipRealTime(e.target, formatted);

} else {

clearValidationFeedback(e.target);

}

});

// Validation finale à la perte de focus

input.addEventListener('blur', function(e) {

if (e.target.value.length > 0) {

validateNipRealTime(e.target, e.target.value);

}

});

// Configuration des attributs

input.setAttribute('placeholder', NIP\_CONFIG.example);

input.setAttribute('maxlength', NIP\_CONFIG.maxLength.toString());

input.setAttribute('pattern', '[A-Z0-9]{2}-[0-9]{4}-[0-9]{8}');

// Ajouter aide contextuelle

addNipHelp(input);

});

console.log('✅ Validation NIP initialisée - Format:', NIP\_CONFIG.format);

}

/\*\*

\* ✅ VALIDATION TEMPS RÉEL AVEC FEEDBACK VISUEL

\* @param {HTMLElement} input

\* @param {string} nip

\*/

function validateNipRealTime(input, nip) {

const validation = validateNipFormat(nip);

const container = input.closest('.form-group, .mb-3, .mb-4, .input-group') || input.parentElement;

// Nettoyer ancien feedback

clearValidationFeedback(input);

if (validation.valid) {

input.classList.remove('is-invalid');

input.classList.add('is-valid');

if (validation.warning) {

showValidationMessage(container, validation.message, 'warning');

} else {

const message = validation.extracted\_info ?

`${validation.message} (Âge: ${validation.extracted\_info.age} ans)` :

validation.message;

showValidationMessage(container, message, 'success');

}

// Validation serveur optionnelle pour vérifier les doublons

validateNipWithServer(input, nip);

} else {

input.classList.remove('is-valid');

input.classList.add('is-invalid');

showValidationMessage(container, validation.message, 'error');

}

}

/\*\*

\* ✅ VALIDATION SERVEUR POUR VÉRIFIER LES DOUBLONS

\* @param {HTMLElement} input

\* @param {string} nip

\*/

async function validateNipWithServer(input, nip) {

try {

const response = await fetch(NIP\_CONFIG.apiUrl, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content

},

body: JSON.stringify({ nip: nip })

});

const result = await response.json();

const container = input.closest('.form-group, .mb-3, .mb-4, .input-group') || input.parentElement;

if (result.success && result.valid) {

if (!result.available) {

// NIP déjà utilisé

input.classList.remove('is-valid');

input.classList.add('is-invalid');

showValidationMessage(container, result.message, 'warning');

} else {

// NIP valide et disponible

const message = `NIP valide et disponible (Âge: ${result.age} ans)`;

showValidationMessage(container, message, 'success');

}

} else if (!result.valid) {

input.classList.remove('is-valid');

input.classList.add('is-invalid');

showValidationMessage(container, result.message, 'error');

}

} catch (error) {

console.warn('Erreur validation serveur NIP:', error);

// Ne pas bloquer l'interface en cas d'erreur serveur

}

}

/\*\*

\* ✅ AFFICHER MESSAGE DE VALIDATION

\* @param {HTMLElement} container

\* @param {string} message

\* @param {string} type

\*/

function showValidationMessage(container, message, type) {

const existingFeedback = container.querySelector('.nip-validation-feedback');

if (existingFeedback) {

existingFeedback.remove();

}

const feedback = document.createElement('div');

feedback.className = 'nip-validation-feedback small mt-1';

feedback.textContent = message;

switch (type) {

case 'success':

feedback.className += ' text-success';

feedback.innerHTML = '<i class="fas fa-check-circle me-1"></i>' + message;

break;

case 'warning':

feedback.className += ' text-warning';

feedback.innerHTML = '<i class="fas fa-exclamation-triangle me-1"></i>' + message;

break;

case 'error':

feedback.className += ' text-danger';

feedback.innerHTML = '<i class="fas fa-times-circle me-1"></i>' + message;

break;

}

container.appendChild(feedback);

}

/\*\*

\* ✅ NETTOYER FEEDBACK DE VALIDATION

\* @param {HTMLElement} input

\*/

function clearValidationFeedback(input) {

input.classList.remove('is-valid', 'is-invalid');

const container = input.closest('.form-group, .mb-3, .mb-4, .input-group') || input.parentElement;

const feedback = container.querySelector('.nip-validation-feedback');

if (feedback) {

feedback.remove();

}

}

/\*\*

\* ✅ AJOUTER AIDE CONTEXTUELLE POUR LE NIP

\* @param {HTMLElement} input

\*/

function addNipHelp(input) {

const container = input.closest('.form-group, .mb-3, .mb-4') || input.parentElement;

if (!container.querySelector('.nip-help')) {

const help = document.createElement('small');

help.className = 'nip-help text-muted form-text';

help.innerHTML = `

<i class="fas fa-info-circle me-1"></i>

Format: <strong>${NIP\_CONFIG.format}</strong>

<br>• XX = 2 caractères (lettres/chiffres)

<br>• QQQQ = 4 chiffres

<br>• YYYYMMDD = Date de naissance

`;

container.appendChild(help);

}

}

/\*\*

\* ✅ VALIDATION COMPLÈTE NIP (CLIENT + SERVEUR)

\* @param {HTMLElement} input

\* @param {string} nip

\* @returns {Promise} Résultat de validation

\*/

async function validateNipComplete(input, nip) {

// 1. Validation côté client

const clientValidation = validateNipFormat(nip);

if (!clientValidation.valid) {

validateNipRealTime(input, nip);

return clientValidation;

}

// 2. Validation côté serveur

const container = input.closest('.form-group, .mb-3, .mb-4, .input-group') || input.parentElement;

showValidationMessage(container, 'Vérification en cours...', 'info');

try {

const response = await fetch(NIP\_CONFIG.apiUrl, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').content

},

body: JSON.stringify({ nip: nip })

});

const serverValidation = await response.json();

if (serverValidation.success && serverValidation.valid) {

input.classList.remove('is-invalid');

input.classList.add('is-valid');

const message = serverValidation.available ?

`NIP valide et disponible (Âge: ${serverValidation.age} ans)` :

'NIP valide mais déjà utilisé';

showValidationMessage(container, message,

serverValidation.available ? 'success' : 'warning');

} else {

input.classList.remove('is-valid');

input.classList.add('is-invalid');

showValidationMessage(container, serverValidation.message, 'error');

}

return serverValidation;

} catch (error) {

console.error('Erreur validation serveur:', error);

showValidationMessage(container, 'Erreur de connexion au serveur', 'error');

return { success: false, valid: false, message: 'Erreur serveur' };

}

}

/\*\*

\* ✅ GÉNÉRER EXEMPLE DE NIP VALIDE

\* @returns {string} Exemple de NIP

\*/

function generateNipExample() {

const prefixes = ['A1', 'B2', 'C3', '1A', '2B', '3C'];

const sequences = ['0001', '1234', '5678', '9999'];

const prefix = prefixes[Math.floor(Math.random() \* prefixes.length)];

const sequence = sequences[Math.floor(Math.random() \* sequences.length)];

// Date aléatoire entre 1960 et 2005

const year = 1960 + Math.floor(Math.random() \* 45);

const month = 1 + Math.floor(Math.random() \* 12);

const day = 1 + Math.floor(Math.random() \* 28);

const dateStr = year.toString() +

month.toString().padStart(2, '0') +

day.toString().padStart(2, '0');

return prefix + '-' + sequence + '-' + dateStr;

}

/\*\*

\* ✅ RÉINITIALISER VALIDATION NIP (pour contenu dynamique)

\*/

function reinitNipValidation() {

initNipValidation();

}

// =============================================

// 🚀 INITIALISATION AUTOMATIQUE

// =============================================

// Initialiser au chargement de la page

document.addEventListener('DOMContentLoaded', function() {

initNipValidation();

});

// Exporter les fonctions pour usage global

window.NipValidation = {

validateFormat: validateNipFormat,

formatInput: formatNipInput,

validateComplete: validateNipComplete,

generateExample: generateNipExample,

reinit: reinitNipValidation

};

**🔧 Étape 2B : Mise à jour du fichier create.blade.php**

<!-- =============================================

🔧 ÉTAPE 2B : MODIFICATIONS DANS create.blade.php

============================================= -->

<!-- 🎯 MODIFICATION 1 : Champ NIP demandeur dans l'ÉTAPE 3 -->

<!-- Chercher cette section vers la ligne 400-450 : -->

<!-- NIP - VALIDATION MISE À JOUR -->

<div class="col-md-6 mb-4">

<label for="demandeur\_nip" class="form-label fw-bold required">

<i class="fas fa-hashtag me-2 text-primary"></i>

Numéro d'Identification Personnelle (NIP)

</label>

<div class="input-group">

<input type="text"

class="form-control form-control-lg"

id="demandeur\_nip"

name="demandeur\_nip"

data-validate="nip"

placeholder="A1-2345-19901225"

maxlength="16"

pattern="[A-Z0-9]{2}-[0-9]{4}-[0-9]{8}"

required>

<span class="input-group-text">

<i class="fas fa-spinner fa-spin d-none" id="nip-loading"></i>

<i class="fas fa-check text-success d-none" id="nip-valid"></i>

<i class="fas fa-times text-danger d-none" id="nip-invalid"></i>

</span>

</div>

<div class="form-text">

<i class="fas fa-info me-1"></i>

Format nouveau: XX-QQQQ-YYYYMMDD (ex: A1-2345-19901225)

</div>

<div class="invalid-feedback" id="demandeur\_nip\_error"></div>

</div>

<!-- 🎯 MODIFICATION 2 : Champs NIP fondateurs dans l'ÉTAPE 6 -->

<!-- Chercher cette section vers la ligne 1200-1250 : -->

<div class="col-md-3 mb-3">

<label for="fondateur\_nip" class="form-label fw-bold">NIP</label>

<input type="text"

class="form-control"

id="fondateur\_nip"

data-validate="nip"

placeholder="A1-2345-19901225"

maxlength="16"

pattern="[A-Z0-9]{2}-[0-9]{4}-[0-9]{8}">

<small class="form-text text-muted">Format: XX-QQQQ-YYYYMMDD</small>

</div>

<!-- 🎯 MODIFICATION 3 : Champs NIP adhérents dans l'ÉTAPE 7 -->

<!-- Chercher cette section vers la ligne 1400-1450 : -->

<div class="col-md-4 mb-3">

<label for="adherent\_nip" class="form-label fw-bold">NIP</label>

<input type="text"

class="form-control"

id="adherent\_nip"

data-validate="nip"

placeholder="A1-2345-19901225"

maxlength="16"

pattern="[A-Z0-9]{2}-[0-9]{4}-[0-9]{8}">

<small class="form-text text-muted">Format: XX-QQQQ-YYYYMMDD</small>

</div>

<!-- 🎯 MODIFICATION 4 : Section scripts à la fin du fichier -->

<!-- Chercher la section @push('scripts') vers la fin et AJOUTER : -->

@push('scripts')

<!-- Scripts existants -->

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

<!-- ✅ NOUVEAU : Validation NIP format XX-QQQQ-YYYYMMDD -->

<script src="{{ asset('js/nip-validation.js') }}"></script>

<!-- Scripts existants -->

<script src="{{ asset('js/workflow-2phases.js') }}"></script>

<script src="{{ asset('js/organisation-create.js') }}"></script>

<script src="{{ asset('js/chunking-import.js') }}"></script>

@endpush

<!-- 🎯 MODIFICATION 5 : Ajouter section d'aide NIP dans les modals -->

<!-- Chercher le modal d'aide vers la fin et AJOUTER cette section : -->

<!-- Modal d'aide NIP -->

<div class="modal fade" id="nipHelpModal" tabindex="-1">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header bg-info text-white">

<h5 class="modal-title">

<i class="fas fa-hashtag me-2"></i>

Aide - Format NIP Gabonais

</h5>

<button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal"></button>

</div>

<div class="modal-body">

<h6>Nouveau format NIP : XX-QQQQ-YYYYMMDD</h6>

<div class="alert alert-info">

<h6 class="alert-heading">Structure du NIP :</h6>

<ul class="mb-0">

<li><strong>XX</strong> : 2 caractères alphanumériques (A-Z, 0-9)</li>

<li><strong>QQQQ</strong> : 4 chiffres de séquence</li>

<li><strong>YYYYMMDD</strong> : Date de naissance (Année-Mois-Jour)</li>

</ul>

</div>

<h6>Exemples valides :</h6>

<div class="bg-light p-3 rounded">

<code>A1-2345-19901225</code> → Né le 25/12/1990<br>

<code>B2-0001-20000115</code> → Né le 15/01/2000<br>

<code>C3-9999-19850630</code> → Né le 30/06/1985

</div>

<div class="mt-3">

<button type="button" class="btn btn-outline-info" onclick="showNipExample()">

<i class="fas fa-magic me-2"></i>Générer un exemple

</button>

</div>

<div id="nipExampleResult" class="mt-2"></div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Fermer</button>

</div>

</div>

</div>

</div>

<!-- 🎯 MODIFICATION 6 : Script inline pour exemples NIP -->

<script>

function showNipExample() {

if (window.NipValidation) {

const example = window.NipValidation.generateExample();

const validation = window.NipValidation.validateFormat(example);

const resultDiv = document.getElementById('nipExampleResult');

resultDiv.innerHTML = `

<div class="alert alert-success">

<strong>Exemple généré :</strong> <code>${example}</code><br>

<small>Âge calculé : ${validation.extracted\_info.age} ans</small>

</div>

`;

}

}

// Ajouter bouton d'aide NIP dans les champs

document.addEventListener('DOMContentLoaded', function() {

const nipInputs = document.querySelectorAll('input[data-validate="nip"]');

nipInputs.forEach(function(input) {

const helpBtn = document.createElement('button');

helpBtn.type = 'button';

helpBtn.className = 'btn btn-outline-info btn-sm ms-2';

helpBtn.innerHTML = '<i class="fas fa-question-circle"></i>';

helpBtn.onclick = function() {

new bootstrap.Modal(document.getElementById('nipHelpModal')).show();

};

const container = input.closest('.input-group') || input.parentElement;

if (!container.querySelector('.btn-outline-info')) {

container.appendChild(helpBtn);

}

});

});

</script>

**Étape 2C : Mise à jour du fichier organisation-create.js**

/\*\*

\* =============================================

\* 🔧 MISES À JOUR DANS organisation-create.js

\* =============================================

\*/

// 🎯 AJOUT 1 : Chercher la fonction de validation des fondateurs et AJOUTER :

/\*\*

\* ✅ VALIDATION FONDATEUR AVEC NOUVEAU FORMAT NIP

\*/

function validateFondateur() {

const nom = document.getElementById('fondateur\_nom').value.trim();

const prenom = document.getElementById('fondateur\_prenom').value.trim();

const nip = document.getElementById('fondateur\_nip').value.trim();

const fonction = document.getElementById('fondateur\_fonction').value;

const telephone = document.getElementById('fondateur\_telephone').value.trim();

// Validation de base

if (!nom || !prenom || !nip || !fonction || !telephone) {

showAlert('Tous les champs obligatoires doivent être remplis', 'warning');

return false;

}

// ✅ VALIDATION NOUVEAU FORMAT NIP

if (window.NipValidation) {

const nipValidation = window.NipValidation.validateFormat(nip);

if (!nipValidation.valid) {

showAlert(`NIP invalide: ${nipValidation.message}`, 'danger');

return false;

}

// Vérifier l'âge

if (nipValidation.extracted\_info && nipValidation.extracted\_info.age < 18) {

showAlert(`Le fondateur doit être majeur (âge détecté: ${nipValidation.extracted\_info.age} ans)`, 'danger');

return false;

}

}

// Vérifier doublons dans la liste actuelle

const fondateurs = getFondateursList();

const nipExists = fondateurs.some(f => f.nip === nip);

if (nipExists) {

showAlert('Ce NIP existe déjà dans la liste des fondateurs', 'warning');

return false;

}

return true;

}

// 🎯 AJOUT 2 : Chercher la fonction de validation des adhérents et AJOUTER :

/\*\*

\* ✅ VALIDATION ADHERENT AVEC NOUVEAU FORMAT NIP

\*/

function validateAdherent() {

const nom = document.getElementById('adherent\_nom').value.trim();

const prenom = document.getElementById('adherent\_prenom').value.trim();

const nip = document.getElementById('adherent\_nip').value.trim();

const profession = document.getElementById('adherent\_profession').value.trim();

// Validation de base

if (!nom || !prenom || !nip || !profession) {

showAlert('Tous les champs obligatoires doivent être remplis', 'warning');

return false;

}

// ✅ VALIDATION NOUVEAU FORMAT NIP

if (window.NipValidation) {

const nipValidation = window.NipValidation.validateFormat(nip);

if (!nipValidation.valid) {

showAlert(`NIP invalide: ${nipValidation.message}`, 'danger');

return false;

}

// Vérifier l'âge

if (nipValidation.extracted\_info && nipValidation.extracted\_info.age < 18) {

showAlert(`L'adhérent doit être majeur (âge détecté: ${nipValidation.extracted\_info.age} ans)`, 'danger');

return false;

}

}

// Vérifier doublons dans la liste actuelle

const adherents = getAdherentsList();

const nipExists = adherents.some(a => a.nip === nip);

if (nipExists) {

showAlert('Ce NIP existe déjà dans la liste des adhérents', 'warning');

return false;

}

return true;

}

// 🎯 AJOUT 3 : Fonction de validation du formulaire complet

/\*\*

\* ✅ VALIDATION FORMULAIRE COMPLET AVEC NOUVEAU FORMAT NIP

\*/

function validateFormStep3() {

const demandeurNip = document.getElementById('demandeur\_nip').value.trim();

if (!demandeurNip) {

showAlert('Le NIP du demandeur est obligatoire', 'danger');

return false;

}

// ✅ VALIDATION NOUVEAU FORMAT NIP DEMANDEUR

if (window.NipValidation) {

const nipValidation = window.NipValidation.validateFormat(demandeurNip);

if (!nipValidation.valid) {

showAlert(`NIP demandeur invalide: ${nipValidation.message}`, 'danger');

document.getElementById('demandeur\_nip').focus();

return false;

}

// Vérifier l'âge du demandeur

if (nipValidation.extracted\_info && nipValidation.extracted\_info.age < 18) {

showAlert(`Le demandeur doit être majeur (âge détecté: ${nipValidation.extracted\_info.age} ans)`, 'danger');

return false;

}

}

return true;

}

// 🎯 AJOUT 4 : Fonction d'aide pour le formatage automatique

/\*\*

\* ✅ INITIALISATION FORMATAGE AUTOMATIQUE NIP

\*/

function initNipFormatting() {

// Initialiser le formatage pour tous les champs NIP

const nipFields = ['demandeur\_nip', 'fondateur\_nip', 'adherent\_nip'];

nipFields.forEach(fieldId => {

const field = document.getElementById(fieldId);

if (field) {

// Formatage en temps réel

field.addEventListener('input', function(e) {

if (window.NipValidation) {

e.target.value = window.NipValidation.formatInput(e.target.value);

}

});

// Validation à la perte de focus

field.addEventListener('blur', function(e) {

if (e.target.value && window.NipValidation) {

const validation = window.NipValidation.validateFormat(e.target.value);

// Afficher feedback immédiat

const feedback = e.target.parentElement.querySelector('.nip-feedback') ||

document.createElement('small');

feedback.className = 'nip-feedback form-text';

if (validation.valid) {

feedback.className += ' text-success';

feedback.innerHTML = `<i class="fas fa-check me-1"></i>${validation.message}`;

e.target.classList.remove('is-invalid');

e.target.classList.add('is-valid');

} else {

feedback.className += ' text-danger';

feedback.innerHTML = `<i class="fas fa-times me-1"></i>${validation.message}`;

e.target.classList.remove('is-valid');

e.target.classList.add('is-invalid');

}

if (!e.target.parentElement.querySelector('.nip-feedback')) {

e.target.parentElement.appendChild(feedback);

}

}

});

}

});

}

// 🎯 AJOUT 5 : Fonction de validation avant soumission

/\*\*

\* ✅ VALIDATION FINALE AVANT SOUMISSION

\*/

function validateAllNipsBeforeSubmit() {

const errors = [];

// Vérifier NIP demandeur

const demandeurNip = document.getElementById('demandeur\_nip').value.trim();

if (demandeurNip && window.NipValidation) {

const validation = window.NipValidation.validateFormat(demandeurNip);

if (!validation.valid) {

errors.push(`NIP demandeur: ${validation.message}`);

}

}

// Vérifier NIP fondateurs

const fondateurs = getFondateursList();

fondateurs.forEach((fondateur, index) => {

if (window.NipValidation) {

const validation = window.NipValidation.validateFormat(fondateur.nip);

if (!validation.valid) {

errors.push(`NIP fondateur ${index + 1} (${fondateur.nom}): ${validation.message}`);

}

}

});

// Vérifier NIP adhérents

const adherents = getAdherentsList();

adherents.forEach((adherent, index) => {

if (window.NipValidation) {

const validation = window.NipValidation.validateFormat(adherent.nip);

if (!validation.valid) {

errors.push(`NIP adhérent ${index + 1} (${adherent.nom}): ${validation.message}`);

}

}

});

if (errors.length > 0) {

const errorMessage = 'Erreurs de format NIP détectées:\n\n' + errors.join('\n');

showAlert(errorMessage, 'danger');

return false;

}

return true;

}

// 🎯 AJOUT 6 : Mise à jour de l'initialisation

// Chercher la fonction d'initialisation existante et AJOUTER :

document.addEventListener('DOMContentLoaded', function() {

// ... code existant ...

// ✅ INITIALISER FORMATAGE NIP

initNipFormatting();

// ✅ MISE À JOUR VALIDATION ÉTAPES

if (typeof updateStepValidation === 'function') {

const originalValidateStep3 = validateFormStep3;

validateFormStep3 = function() {

return originalValidateStep3() && validateFormStep3();

};

}

// ✅ MISE À JOUR SOUMISSION FORMULAIRE

const form = document.getElementById('organisationForm');

if (form) {

form.addEventListener('submit', function(e) {

if (!validateAllNipsBeforeSubmit()) {

e.preventDefault();

return false;

}

});

}

console.log('✅ Validation NIP intégrée dans organisation-create.js');

});

// 🎯 AJOUT 7 : Fonctions utilitaires pour les listes

/\*\*

\* ✅ OBTENIR LISTE DES FONDATEURS

\*/

function getFondateursList() {

// Cette fonction doit retourner la liste actuelle des fondateurs

// Adapter selon votre implémentation existante

const fondateursData = localStorage.getItem('fondateurs');

return fondateursData ? JSON.parse(fondateursData) : [];

}

/\*\*

\* ✅ OBTENIR LISTE DES ADHERENTS

\*/

function getAdherentsList() {

// Cette fonction doit retourner la liste actuelle des adhérents

// Adapter selon votre implémentation existante

const adherentsData = localStorage.getItem('adherents');

return adherentsData ? JSON.parse(adherentsData) : [];

}

/\*\*

\* ✅ AFFICHER ALERTE

\*/

function showAlert(message, type = 'info') {

// Adapter selon votre système d'alertes existant

if (typeof Swal !== 'undefined') {

Swal.fire({

icon: type === 'danger' ? 'error' : type,

title: type === 'danger' ? 'Erreur' : 'Information',

text: message

});

} else {

alert(message);

}

}

📋 **Guide d'implémentation Frontend - Actions à effectuer :**

Parfait ! Passons maintenant à l'implémentation du frontend JavaScript pour le nouveau format NIP.

**🚀 Étape 2 : Frontend JavaScript - Validation NIP XX-QQQQ-YYYYMMDD**

**🔧 Étape 2A : Création du fichier de validation NIP### 🔧 Étape 2B : Mise à jour du fichier create.blade.php### 🔧 Étape 2C : Mise à jour du fichier organisation-create.js## 📋 Guide d'implémentation Frontend - Actions à effectuer :**

**🔧 Étape 2A : Créer le fichier de validation NIP**

1. **Créer le fichier :** public/js/nip-validation.js
2. **Copier** tout le contenu du premier artifact JavaScript
3. **Vérifier** que le fichier est bien dans public/js/

**🔧 Étape 2B : Modifier create.blade.php**

**Chercher et remplacer ces sections :**

1. **Champ NIP demandeur** (Étape 3) - Ligne ~400-450 :
2. <!-- REMPLACER le champ demandeur\_nip existant -->
3. **Champ NIP fondateur** (Étape 6) - Ligne ~1200-1250 :
4. <!-- REMPLACER le champ fondateur\_nip existant -->
5. **Champ NIP adhérent** (Étape 7) - Ligne ~1400-1450 :
6. <!-- REMPLACER le champ adherent\_nip existant -->
7. **Section scripts** - Fin du fichier dans @push('scripts') :
8. <!-- AJOUTER cette ligne -->
9. <script src="{{ asset('js/nip-validation.js') }}"></script>

**🔧 Étape 2C : Modifier organisation-create.js**

**Fichier :** public/js/organisation-create.js

**Ajouter les nouvelles fonctions** à la fin du fichier existant (avant la dernière });)

**🔧 Étape 2D : Test rapide**

**Ouvrir la page de création et vérifier :**

1. **Champ NIP se formate** automatiquement pendant la saisie
2. **Validation temps réel** avec messages colorés
3. **Exemples valides :**
   * A1-2345-19901225
   * B2-0001-20000115
   * C3-9999-19850630

**✅ Fonctionnalités implémentées :**

* ✅ **Formatage automatique** : XX-QQQQ-YYYYMMDD
* ✅ **Validation temps réel** côté client
* ✅ **Validation serveur** pour doublons
* ✅ **Calcul automatique de l'âge** depuis le NIP
* ✅ **Détection mineurs** (< 18 ans)
* ✅ **Messages d'aide contextuels**
* ✅ **Exemples générés automatiquement**

**Avez-vous des questions sur l'implémentation ou êtes-vous prêt à tester ces modifications ?**